21.01.25

// LÓGICA DE PROGRAMAÇÃO EM PYTHON
CAPACITA BRASIL

AULA 02 - Introdução à lógica de Programação em python
- Linguagem de Programação;
- Ambientes de Programação Python;
- Operações Matemáticas;

(A) Noções de Linguagens de Programação e sua Importância
* O que são linguagens de programação?
  - Instrumento para transmitir instruções → Computadores;
  - ~~Cada tipo~~ Cada linguagem tem um propósito / objetivo;

JAVA:
- Linguagem de programação / plataforma de Computação;
- Sun 1995;
- Orientada a objetos;
- Versátil e poderosa;
- De softwares a app;

PHP:
- Hypertext Preprocessor;
- Desenvolvimento web;
- Wordpress, Facebook; (Comércio eletrônico / Redes sociais)

C++:
- Propósito geral;
- Extensão da C;
- Áreas da tecnologia;
- Desenvolvimento de sistemas; (Embarcados)

(B) Linguagem de Programação python - Propósito Geral;
Ex:
- Desenvolvimento web; (Frameworks / Bibliotecas) Django / Flask
- Ciências de dados; (SciPy) (Matplotlib) (Pandas) ← Bibliotecas
- Machine learning;
- Alto nível - 1991;

- Principal aplicação
- Análise de dados
+ Bibliotecas específicas.

- Sintaxe clara e legível;
- Multiplataforma: diferentes sistemas;
- Comunidade ativa;
- Jogos: Pygame para desenvolver game;

© Preparação do Ambiente de Programação

Continuação

1. Conceitos básicos de algoritmos e lógica de programação

Software
- Programas de computador;
- Comuns, Lazer. Trabalho. Estudo;
- Redes sociais. Jogos.

2. Lógica de Programação

Conjunto de passos escrito em uma linguagem de programação.

Algoritmos
- ▷ procedimento que leva em consideração dados de entrada (inputs) e produz dados de saída (outputs)
- ▷ Sequência de passos computacionais que transformam entradas (inputs) em saídas (outputs).

Dados → (Entrada) → (Processamento) → (Saída)

Operação: Processamento; Resultado: Saída

2. Noções de linguagem de programação e sua importância

- Linguagem de programação: instrumentos que permitem transmitir instruções aos computadores de maneira clara e eficaz.

▷ Cada um tem um propósito específico para sua aplicação.

02. Variáveis
1. Crie seus primeiros códigos
2. Variáveis
3. Estruturas de Dados

02.02.25

Aula 01: Introdução à Programação Visual

01. Tipos de Variáveis

* Variáveis: funcionam como espaço para armazenar dados;
  - São usadas por todas as linguagens de programação;
  - Representados por letras ou substantivos.

Ex: a = 2
b = 3

  - Podem armazenar diferentes tipos de dados:
    - Inteiros (int)
    - Float (Ponto flutuante)
    - String (str)
    - Booleano (boolean)

02. Operações com Variáveis

- Operadores: (+) / (-) / (*) / (**) Potenciação / (/) divisão
- Operações com Strings
  - Concatenação (+): União de String;
  - Repetição (*):

03. Estrutura de Dados

- Maneiras organizadas e eficientes de armazenar, organizar e manipular dados. São fundamentais para a criação de algoritmos eficientes e para a resolução de problemas.
  - Listas (lists) ([~])
    - Coleção mutável de elementos; (Podem conter diferentes tipos)
  - Tuplas (Tuples) ((—))
    - Coleção imutável de elementos. (Não Podem conter diferentes tipos.)

04: Exemplos de Algoritmos

AULA 04 - Estruturas de Repetição e Estruturas de Dados

- Operadores relacionais lógicos
- Estruturas de Repetição
- Estruturas de Dados

Operadores relacionais

<, >	- Manipulação de dados;
<=, >=	- Implementação de lógica;
==, !=	- comparam ou relacionam variáveis;

Operadores lógicos:

- or (ou A ou B) - True;
- and (a e b) - True;
- not

Estruturas de Repetição (loops.)

- Executa um bloco várias vezes;
  - for;
  - while;

Ex:
```
for _ in _:
    print (fruta)
```

* Não sabemos quantas vezes vamos repetir.

```
for _ in range(5):
    print(i)
```
Sabemos o número de repetições repetidas;

* while: Executa um bloco de código enquanto uma condição for (V)

Ex:
```
contador = 5
while contador > 0:
    print()
    contador -= 1
```

CADERSIL
www.cadersil.com.br

```
soma = 0
numero = 1
while numero <= 100:
    soma += numero
    numero += 1
    print("X")
```

- Em Python, tudo é objeto!
- Objetos são instâncias de classes.

- Estruturas de Dados:
  - Listas;
  - Coleções mutáveis de elementos;
  - Dicionários;
    - array associativos
      um valor pode ser vinculado a

AULA 03: Estrutura de Decisão e Repetição

1. Estrutura de decisões:
   - permitem que o fluxo do programa siga diferentes caminhos;
   - dependentes de uma condição específica;
   - muito comuns;
   - * (if): Seleciona qual trecho deve ser executado com algumas condições.
   - = (if - else): Se verdadeiro faz uma coisa;
     Se falso faz outra;
   - = (elif): Verifica condições de forma sequencial;
     Se anterior falhar vai pro elif;

2. Operadores relacionais e lógicos
   - Símbolos específicos que executam operações em variáveis e valores;
   - Manipulação de dados e implementação de lógica;

Relacionais:

< menor / > maior / <= / >=)
== igual / != diferentes

CADERSIL

- Operadores lógicos (or / and / not)
  - Resultado (V/F);
  -

* Estruturas de Repetição (loops)
  - Usados para executar um bloco de dados várias vezes;
  - ~~Ptdo~~ python: for / while;

for: usado para percorrer uma sequência.
  ~~sat~~ ideal: ~~repetir quantas vezes~~ quando você já sabe quantas vezes precisa repetir um bloco de código.
  Range: cria uma lista de números;

While: Executa um bloco de ~~g~~código enquanto uma condição for V;

4. Exemplos
  - Contador;
  - Cálculo de Imposto;

Aula 04: Função e Modularização

1. Funções
  - Subprogramas que tem uma função específica dentro do programa;
  - Podem ser reutilizados;
  - Permitem modularizar o programa;

* Declaração de uma função: processo de definir uma função com nome e/ou parâmetro e um corpo (bloco de código).

* Passagem de parametros: fornecer argumentos para a função;

- Nome: identifica a função. Usado para chamá-la;
- Parametros: variáveis que a função espera ~~receber~~ receber. (Opcionais)
- Corpo: instruções a ser executado;
- Retorno: valor do processamento;

25 · 02 · 25

LPC

* Princípios de Pensamento Computacional com Python

- Introdução
- Pensamento Computacional
    - Pensamento Computacional
    - Qual a importância?
    - Por que aprender com Python?
- Abstração
    - Definição e Exemplos
    - Exemplo prático em Python.

④ Pensamento algorítmico
- Entendendo o que é a lógica;
- Exemplo prático em Python

⑤ Decomposição.
- Definição e Exemplos
- Exemplo prático em python.

⑥ Identificação de Padrões
- Definição e Exemplos
- Exemplo prático em Python.

⑦ Encerramento
- Encerramento
- Materiais de Apoio
- Quiz

PENSAMENTO COMPUTACIONAL COM PYTHON

@ renadeveloper

AULAS
1. Introdução ao Curso
2. Pensamento Computacional
3. Abstração
4. Pensamento Algorítmico
5. Decomposição
6. Reconhecimento de Padrões

OK: 26/02/25
FINALIZADO!

AULA 01 - Introdução
- Apresentar conceitos de Pensamento Computacional; (Objetivo)

AULA 02 - P.C.
- O que é P.C.?

NÃO É: 1. Mexer no Computador;
2. Preciso usar Computador;
3. Só para quem é da área;

DEF.: (WING, 2014) "São processos de pensamento envolvidos na formulação de um problema e que expressam sua solução ou soluções eficazmente, de tal forma que uma máquina ou uma pessoa possam realizar."

(FURBER, 2012) "Processo de reconhecer aspectos da computação em um mundo que nos cerca e, aplicar ferramentas e técnicas da ciência da Computação para entender e argumentar sobre sistemas e processos naturais e artificiais.

▷ Resolver problemas com ~~asspectos~~ aspectos comuns a computação!

PILARES:
- ABSTRAÇÃO
- DECOMPOSIÇÃO
- PENSAMENTO ALGORÍTMICO
- IDENTIFICAÇÃO DE PADRÕES

▷ Importância
- Letramento Digital; (Mundo Digitalizado)
- Resolução de Problemas;
- Tomada de decisão;
- Tecno-pedagogia;
-

* Por que aprender com Python?
  - linguagem de propósito geral; (web, dados, app...)
  - Criada: Guido van Rossum;
    - 1989;
    - lançada 1991; (primeiros scripts)
  - + Cresceu nos últimos 5 anos;
  - Curva de aprendizado alta; (Escalável)
  - Muitas bibliotecas;
  - Comunidade forte;
  - + Popular;
  - Passou JAVA 2019;

> FOCAR NO MAIS IMPORTÂNTE!

P.01 - ABSTRAÇÃO: Ação de ignorar os detalhes de uma solução de modo que ela possa ser válida para diversos problemas.

Isolar aspectos relevantes de um todo e considerá-lo de forma individual.

▷ Neste ponto pouco me importa os meios, mas sim o fim.

* Pensar no que é. Detalhes para depois!
* Abstrair + Importante;

Ex: $\frac{nota 1 + nota 2}{2}$ => Média

▷ Podemos não saber qual método utilizado para conseguir a nota, mas abstraímos isso e extraímos a informação mais importante: a nota!

Abstraímos essa etapa.

▷ Não nos interessa como essas notas foram obtidas!

26-02-25

P02. PENSAMENTO ALGORÍTMICO

A. Entendendo o que é Lógica
- O que são algoritmos e pseudocódigo;

O QUE É?

Def: Ação de pensar a solução de um problema a partir de uma sequência finita de passos

- CÓDIGO: descrevemos passo-a-passo o que ele faz. (Deve fazer)
- Testar com "teste de mesa."; (Papel) (Passo-a-Passo)

Caso não resolva o problema.

- Vários possíveis soluções; (Podemos usar abstração.)

"Não nos importamos com detalhes nesse momento"

DIVIDIR PARA CONQUISTAR!

P03 DECOMPOSIÇÃO

A. Def: Ação de dividir um problema grande em partes menores.

LÓGICA ▷ Resolver as partes do problema para então obter uma resposta do problema maior.

COMPUTAÇÃO: - Programas grandes;
- Dividimos a escrita em partes menores;
- Aglutinamos tudo no final;

PARTES DO PROGRAMA

Resolvendo cada parte temos a solução do todo.

P04 Identificação de Padrões:

A. Def: Ação de descrever o que vai acontecer com base em eventos anteriores.

▷ Identificar padrões na estrutura!

COMPUTAÇÃO: - Estruturas de repetição;
- Blocos de código semelhantes;

CADERSIL www.cadersil.com.br

26 · 02 · 25

DIO (Digital Innovation One)

▷ INTRODUÇÃO À PROGRAMAÇÃO E PENSAMENTO COMPUTACIONAL

A.1. Pensamento Computacional
A.2. Introdução à lógica de programação
A.3. Fundamentos de algoritmos
A.4. Linguagem de Programação
A.5. Primeiro Contato com a Programação

O QUE É PENSAR COMPUTACIONALMENTE?

01. Pensamento Computacional

- Apresentação;
- Introdução;
- Habilidades complementares;
- Pilares: Decomposição
- Pilares: Padrões
- Pilares: Abstração
- Pilares: Algoritmos

- Estudo de caso conceitual: perdido
- Estudo de caso aplicado: Soma de um intervalo;
- Estudo de caso aplicado: Adicionando um número;

Resolução de problemas!

▷ Primeiros passos para começar a programar!

A. Apresentação

@in/juliana-mascarenhas-ds/

- Modelagem de Computacional (LNCC)
- Data Science
- Java/Python

UBUNTU

▷ Javascript
Python

* Big Data!

* Maneira de Estruturar o Pensamento!

- Aplicável à qualquer área do conhecimento;
- Resolver problema de forma mais objetiva e eficiente;

AULA 01: Pensamento Computacional (P.C.)

A. Introdução: apresentar conceitos que caracterizam o pensamento computacional.

- P.C Não está restrito ao meio computacional;

OVERVIEW

P.C. - "Processo de pensamento envolvido na expressão de soluções em passos computacionais ou algoritmos que podem ser implementados no computador." (Lee, 2016)

P.C.: Sistemático / Eficiente;
- Formulação de resolução de problemas;
- Máquinas e humanos: capazes de resolver;
- Habilidade generalista;
- Matemática / Leitura / Escrita;

(A) Os 4 pilares:
- Decomposição - Separar em partes menores;
- Reconhecimento de Padrões - Identificar padrões ou tendências;
- Abstração
- Design de Algoritmo

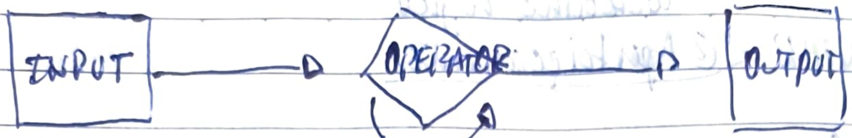
* DECOMPOSIÇÃO
- Pesquisa binária;
- Quebra cabeça;
- Por tipo / Cor.

* Reconhecimento de Padrões:
- Similaridade e diferenças entre os problemas. (Tem diferença de contexto)
- Padrão Comportamental;

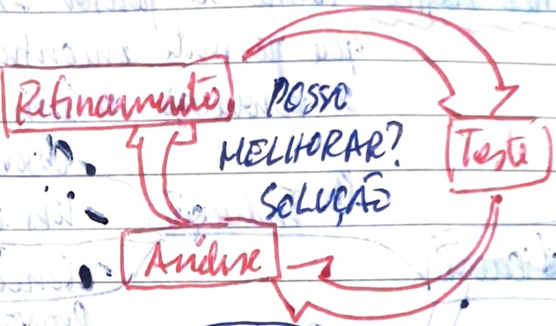
* NÃO USAMOS TODOS AO MESMO TEMPO!

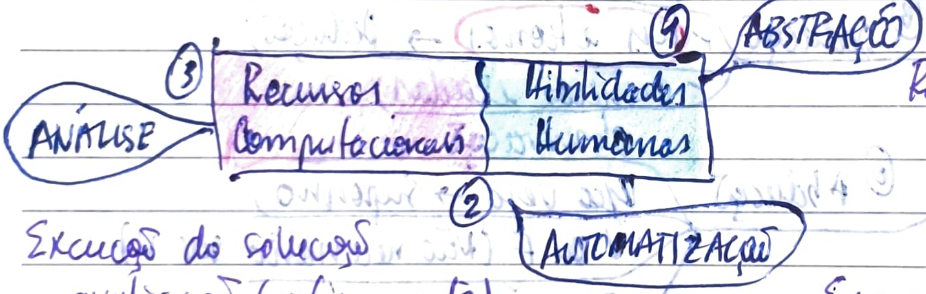
* ABSTRAÇÃO
- Extrapolar o conceito do problema para uma forma generalista;
* Se parametrizar → consigo Generalizar;
/ Particular → Generalista.

* DESIGN DE ALGORITMO
- Automatizar;
- Definir passo-a-passo a solução do problema;
- Instruções;



- Processo Contínuo;
- Definir uma solução;
- Testar a solução;
- Aperfeiçoamento da solução;





Resolução de Problemas

Execução da solução e avaliação (refinamento)

Expressão da solução

* USO D

Ex: Aperfeiçoamento de reações químicas com algoritmos;
- Simuladores em softwares;
- Simulação de problemas em computadores;

- Raciocínio lógico;
- Decomposição;
- Padrões;
- Abstração;
- Algoritimo;
- Refinamento;

* COMPETÊNCIAS:
  - Pensamento Sistemático;
  - Colaboração em equipe;
  - Criatividade e Design;
  - Facilitador;

27 · 02 · 25

HABILIDADE COMPLEMENTAR
- ① Raciocínio Lógico
- ② Aperfeiçoamento

① **Raciocínio Lógico**: forma de pensar estruturadamente, ou raciocínio, que permite encontrar a conclusão ou determinar a resolução de um problema.

VEJO → DEDUZO
VEJO → ENTÃO ??

Classificação
- Ⓐ Indução
  - Observação!
  - Leis e teorias;
  - ciências experimentais (Empirismo)
- Ⓑ Dedução
  - Provar
  - Leis e teorias → Deduções;
  - ciências exatas;
- Ⓒ Abdução
  - observação
  - Vejo → Suponho;
  - V ou F (Não necessário ser V)
  - Diagnóstico
  - Processo investigativo

Inferência
- Sintética (por construção)
  - Abdução (por conclusão)
  - Indução (por experimentação)
- Analítica (por análise)
  - Dedução (por observação)

② **Aperfeiçoamento**: A partir de uma solução determinar pontos de melhora e refinamento.
- Melhoramento
- Aprimoramento
- Refinamento
- Ato de aperfeiçoar

CADERSIL
www.cadersil.com.br

CÓDIGO LIMPO!

* Melhorar tudo ou apenas parte do todo!

A) Melhoramento de recursos:
- Soluções + eficientes;
- Otimizar processos;
- Libera mão de obra;

Melhorar códigos e algoritmos
- Simplificar linhas de códigos;
- Espaços bem definidos;
- Nomes de variáveis bem definidos.

▷ Modularização:
- dividir um programa extenso em partes menores. (Módulos).
- Melhora a manutenção;
- Facilita a reutilização;
- Organiza o projeto de forma eficiente;
- Implementados: arquivo.py;

Exemplos:
- Encontrar máximo da lista;
- Calcular média em uma lista;
- Ordenar lista;

www.cadersil.com.br

CADERSIL

01 · 03 · 25

TÓPICOS AVANÇADOS (PYTHON)

01. List Comprehension

DEF: Forma concisa e eficiente de criar listas a partir de iteráveis. ▷ (for in line)

POSSO USAR: Listas, strings, range...
- Aplicando condições ou transformações.
- Apenas 1 linha
- Realizar operações simples dentro de uma linha.

Usamos menos linhas!

* Operações simples gerando outra lista;

* Sintaxe: [expressão (for) elemento (in) iterável (if) condição]

▷ (for in line) for em uma linha.

Ex: quadrados = [x**2 for x in range(1,6)]
(x**2: EXPRESSÃO; x: ELEMENTO; range(1,6): ITERÁVEL)

Ex: pares = [x (for) x (in) range(1, 11) (if) x % 2 == 0]
(x: EXP.; x: ELEM.; range(1, 11): ITERÁVEL; x % 2 == 0: CONDIÇÃO)

02. Enumerate (Função embutida)

- Retorna um objeto enumerado.
- Gera TUPLAS: (contador (índice); valor do iterável)

* Usamos quando precisamos acessar: / o índice; (e) / valor do elemento. | índice : Valor |

▷ **IMPORTANTE:** itera sobre um iterável, obter o índice e o valor; (/elemento)

* Itera cada elemento da lista; (start)

Sintax: for índice, elemento in enumerate (iterável, início=0):

▷ Altera o índice inicial

Ex:

```
frutas = ['maçã', 'banana', 'uva']
    for indice, fruta in enumerate(frutas):
    print(f"{indice}: {frutas}")
```

CADERSIL www.cadersil.com.br

* Lambda cálculo
* Programação funcional

03. Reduce: → Aplica uma função cumulativa a todos os elementos.
- processa dados iterativamente;
- evita loops explícitos;
- biblioteca: functools;
- Reduz o iterável a um único valor;

Sintaxe: from functools import reduce
reduce (função, iterável)

Lambda: An anonymous inline function of a single expression, which is evaluated when is called.
Sintax: lambda [parameters]: expression

Ex:
```
from functools import reduce
numeros = [1, 2, 3, 4, 5]
soma = reduce (lambda x, y: x+y, numeros)
```
"função oculta"
```
print (soma)
```

Ex:
```
maior = reduce (lambda x, y : x if x>y else y, numeros)
```

* Pode trabalhar com todos da lista

04. Função MAP: Aplica uma função a cada elemento de um iterável!
- Função built-in;
- Permite aplicar uma função a todos os itens de uma sequência;
- Retorna uma nova sequência;
- Retorna um iterador com os resultados;

Sintax
```
map (função, iterável)
```

Ex:
```
numeros = [1, 2, 3, 4]
dobrados = list(map(lambda x: x*2, numeros))

numeros = [10, 20, 30]
strings = list (map (str, numeros))
```

06 · 03 · 25

PENSAMENTO COMPUTACIONAL

* PILARES: Decomposição
  - Problema Complexo: quebram em problemas menores;

"Se temos um problema complexo, temos problemas menores e mais simples para resolver."

(Quebra)

Estratégia: quebrar e determinar partes menores e gerenciáveis. → ANÁLISE → SÍNTESE

ANÁLISE:
- Estudar, explorar;
- Realizar exame;
- Decompor elementos;

* Partes tem que fazer sentido!

Combinar os elementos, recompondo o problema original.
- Reconstrução
- Reunir elementos distintos em um único;

SÍNTESE!

ORDEM
1. Sequencial (Existe dependência entre as partes)
2. Paralela (Tarefas executadas ao mesmo tempo).

- Existem distintas maneiras para decompor problemas; (by your self)

02. Como

Identificar ou coletar Dados > Agregar os Dados > Funcionalidade (Resolução) | DECOMPOSIÇÃO

Ex: Cozinhar

* Funcionamento do Sistema
  - Identificar os componentes;
  - Papel de cada componente;
  - Interdependência das peças;

CADERSIL
www.cadersil.com.br

Ex:

- Criar App
  - Finalidade;
  - Interface;
  - Funcionalidades;
  - Pré-requisitos;

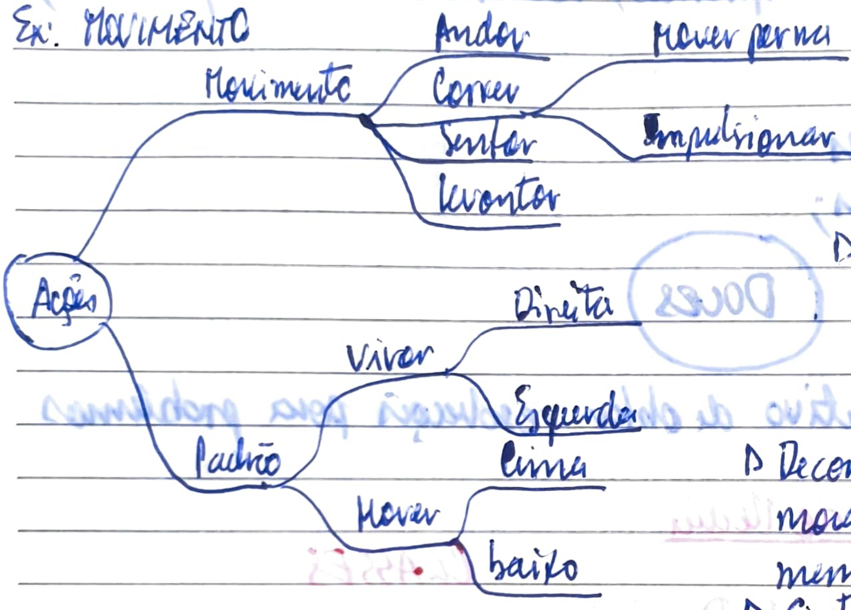
"Definição de Componentes e Etapas"

⊕ Eficiência;

▷ Sequêncial ou Paralela?

▷ Ordem ou não?

Ex: MOVIMENTO



▷ Cada etapa podemos chamar uma função;

▷ Se for preciso modificação iremos mudar apenas pontos chave de cada função.

▷ Decompus (Analisei a Ação, movimento, em várias funções menores.

▷ Sintetizei tudo para gerar a ação.

08.03.25

* PILAR: Padrões (Reconhecimento de Padrões)
    01. Modelo base;
    03. Estrutura invariante;
    04. Repetição;

Similaridades
Diferenças

Ex. Aplicação:
- Compreensão/reconhecimento de padrão;

- Seres vivos x Padrões
    - Com similaridades;

Ex: FRUTAS   DOCES

MOTIVO:
= Generalizar, com objetivo de obter resoluções para problemas diferentes:

CATEGORIAS
- Tipo de Medida
- Depende do Domínio

CLASSES

* HUMANO (Subjetivo)
    - Grau de similaridade;
    - Grupos conhecidos x Objeto desconhecido

* COMPUTADOR (Objetivo / Direto)
    - Comparação

Simular comportamento:
    - Representação de atributos;
    - Aprendizado - Conceito associado ao objeto;
    - Armazenar dados;
    - Regras de decisão;

Extração de Características
Classificação de Dados

→ ABORDAGEM

Diferentes
- Métodos;
- Aplicações;
- Contextos

Aplicações:
- Classificação de dados;
- Reconhecimento de Imagens;
- Reconhecimento de Fala;
- Análise de cenas;
- Classificação de documentos;

→ Machine Learning
- Redes Neurais
- IA
- Ciência de Dados

* PILAR: Abstração

01. Abstrair: Observar, um ou mais elementos, avaliando características e propriedades em separado

- Generalização / Abstração
↳ Processo intelectual de isolamento de um objeto da realidade.

- Generalizar
▷ Torna-se geral, mais amplo, extensão.

Operação intelectual que consiste em reunir numa classe geral, um conjunto de seres ou fenômenos similares

* Classificação:
- Características
- Pontos essenciais
- Generalizar + Detalhar

→ REPRESENTAÇÃO
- Pontos essenciais
- Detalhes

- Conceitos baseados em abstração
  - Merge sort;
  - Clustering;
  - Busca binária;

EXEMPLOS
- Árvore;
- Lista;
- Grafos;

* Tornar ~~proble~~ soluções reutilizáveis

* Linguagens de Programação

* PILAR: Algoritmos

01. Algoritmos
  - Instruções detalhadas;
  - Computador: recebe, manipula e armazena dados;
  - Programas → Instruções;
    - ↳ (Step by step)

▷ Processo de resolução de problemas passo-a-passo utilizando instruções

Algoritmo ⇐ Entendido por Humano e Computador

O QUE PRECISA SER FEITO? → INSTRUÇÕES

QUAL A ORDEM DE EXECUÇÃO? → INSTRUÇÕES

Desenvolvimento do programa:
- Análise: Estudo e definição dos dados de entrada/saída.
- Algoritmo
  - ↳ Descreve o problema por meio de ferramentas:
    - Narrativas;
    - Fluxograma;
    - Pseudo código

CADERSIL www.cadersil.com.br

- Codificação:
  Algoritmo é codificado de acordo com a linguagem de programação

INSTRUÇÕES DETALHADAS ⇒ ALGORITMO ⇒ PROGRAMA

INSTRUÇÕES EXECUTADAS PASSO A PASSO PARA CONCLUIR A TAREFA

- Sequência de passos com objetivo definido;
- Execução de tarefas específicas;
- Conjunto de operações que resultam em uma sucessão finita de ações;

Ex: Trocar uma lâmpada;
- Passos;

* Como construir um algoritmo?
- Compreensão do problema;
- Definição dados de entrada;
- Definir processamento;
- Definir dados de saída;
- Utilizar um método de construção;
- Teste e diagnóstico;

* Construção:
1. - **Narrativa**: - linguagem natural;
   - Diversas interpretações;
2. - **Fluxograma**: - estrutura gráfica;
   - Símbolo definido;
3. - **Pseudocódigo**: - Regras definidas;
   - Portugol;
   - Segue passo a passo;

EXEMPLO

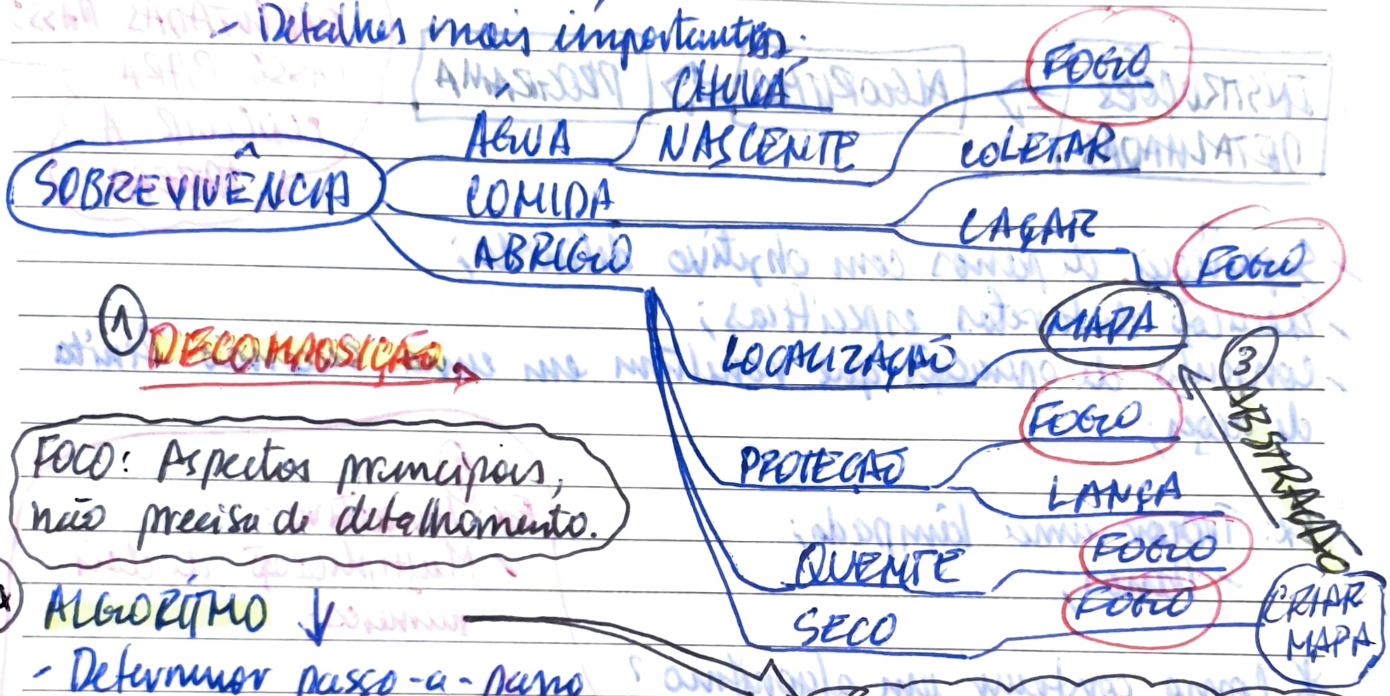
1. Ex: Narrativa:
   - Multiplicação de dois números;
2. Fluxograma:
   INI → N1, N2 → M = N1 · N2 → M → FIM
3. PSEUDOCÓDIGO:
   P1: Recebe valores
   P2: Multiplica
   P3: Imprime Resultado

08 · 03 · 25

* Estudo de Caso: conceitual: perdido na floresta

- Pensamento computacional
    - Identificar mecanismos;
    - Recursos comuns;
    - Detalhes mais importantes.

② PADRÃO



① DECOMPOSIÇÃO

FOCO: Aspectos principais, não precisa de detalhamento.

④ ALGORITMO ↓

- Determinar passo-a-passo para comida:
    - Pegar peixe;
    - Colocar água na panela;
    - Ferver a água;
    - Limpar peixe;
    - Fazer o cozido
    - Ascar;

* Um algoritmo para cada etapa de atividade.

* Estudo de Caso: Soma de um intervalo

* Estudo de Caso: Adivinhe um número

② Introdução à Lógica de Programação

A.01 - O que é Lógica?

A.02 - Técnicas de lógica de programação.

01. O que é Lógica?

Lógica → Forma de raciocínio
Lógica → Organização coesa

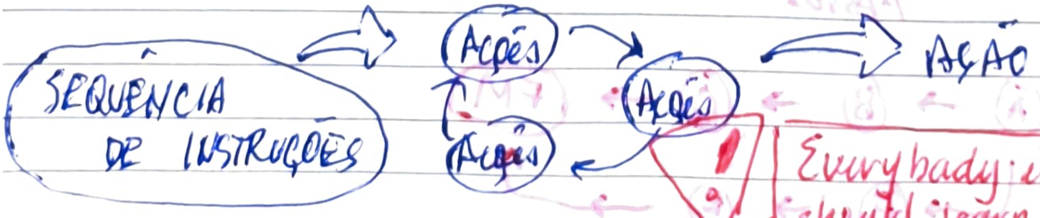
Problema → Questão
Problema → Proposta duvidosa
Problema → Várias Soluções
Problema → Objeto de discurso

* Processo de pensamento atrelado ao conceito de algoritmos e resolução de problemas!

▷ "Questão que foge a uma determinada regra, desvio de percurso o qual impede de atingir um objetivo com eficácia e eficiência."

LÓGICA: "Parte da filosofia que trata das formas do pensamento em geral (dedução, indução, hipótese, inferência etc.) e das operações intelectuais que visam à determinação do que é verdadeiro ou falso."

▷ Organização e planejamento das instruções, assertivas em um algoritmo, a fim de viabilizar a implantação de um programa.



Everybody in this country should learn how to program a computer because it teaches you how to think.

* Humanos - prevê comportamentos;
* PC's - Não prevê;
→ (Instruções detalhadas)!

▷ Técnicas de Lógica de Programação

01. Técnicas

(Planta Baixa) ▶ (Projeto) ▶ (Planejamento) — Segue a lógica dos passos!

Ⓐ Linear: Modelo tradicional → (Modelos de desenvolvimento e resolução.)

Não tem vínculo
- Estrutura hierárquica;
- Programação de computadores;
- Execução sequenciada;
- Recurso limitados;
- Única dimensão;

PASSOS

(INÍCIO) → Ⓐ → Ⓑ → Ⓒ → Ⓓ → (FIM)

(Ordenação de Elementos por uma única propriedade.)

Ⓑ Estrutura: Organização, disposição e ordem dos elementos essenciais que compõem um corpo (concreto ou abstrato)

ESTRUTURADA ▷

OBJETIVO →
- ESCRITA ▷ Programas
- ENTENDIMENTO
- VALIDAÇÃO
- MANUTENÇÃO ← Facilitar

/ Mais de uma hierarquia;

PASSOS

(INÍCIO) → Ⓐ → Ⓑ → Ⓒ → (FIM)

Ⓐ → Ⓔ → Ⓕ → (FIM)

Ⓒ Modular: controlada por um conjunto de regras

(Módulo 1) (Módulo 2) (Módulo 3) ⇒ Partes independentes

CADERSIL

www.cadersil.com.br

* Técnica Modular

Dados de entrada | Processo de transformação | Dados de saída

METAS: - Simplificação;
- Decompor o problema;
- Verificação do módulo;

▷ * Cada ação um módulo!

③ FUNDAMENTOS DE ALGORITMOS

01. Tipologia e variáveis;
02. Instruções primitivas;
03. Estruturas condicionais e operadores;
04. Estruturas de repetição;
05. Vetores e matrizes;
06. O que são funções?
07. Instruções de entrada/saída;

Conceitos como:
- variáveis;
- tipos de dados;
- instruções;
- condições

01. Tipologia e variáveis

- Variáveis;

Função do computador → Processar dados!

Informação → Dados
Informação → Instruções (Operações)

(Tratados e processados!)

Dados:
- Numéricos
- Caracteres
- Lógicos

} primitivos

- Numérico
  - Inteiros (1, 2, -1)
  - Reais (1,35; 2,05)

- Caractere

- Lógico
  - V (1)
  - F (0)

Booleano

CADERSIL

- Variável: irá receber um tipo de dado, mas não sabe o valor.
  - pode assumir qualquer um dos valores de um determinado tipo.
  - Nome:
    - Atribuição de um ou mais caractere;
    - Sem espaços
    - Primeira letra - não número;
    - Vedado: uso de palavra reservada;
    - Caractere e números;

Papel → Ação: Modifica o estado

Controle: Controle dentro de uma equação;

- Constante: Inalterável;
  O que não muda;
  Invariável;
  * Fixo ou estável;

(-)	Subtração
(+)	Adição
(*)	Multiplicação
(div)	divisão
(/)	Divisão
(↑)	Exponenciação

02. Instruções Primitivas → Binários

Cálculo Mat.
Variáveis
Constantes → Operadores → Binários / Unários

Unários:
(+) - Manutenção de Sinal
(-) - Inversão de Sinal

Instruções Primitivas: linguagem de palavras-chave de uma determinada linguagem de programação que tem por finalidade comandar um computador que irá tratar os dados.

// Notação → Operações
- Entrada, processamento e saída; (Dados)

03 - Estruturas condicionais e operadores

- Estrutura condicional (E.C.)
  * Condição - Estado de uma pessoa ou coisa.
  * Condicional -
    - Que expressão uma condição ou suposição;
    - Contém ou implica uma suposição ou hipótese;

Condição → (SATISFEITA) → Operação
↓ (!VERDADE)
Exceção

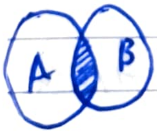
* Simples: Condição → Operação
* Composta: condição → Exceção / → Operação
* Encadeada: Condição → Operação
  ↓
  Condição → Operação / → Exceção

*Operadores Relacionais
=
<>
>
<
>=
<=

*Operadores lógicos
- And
- Or
- Not

▷ Quando usar:
- Verificação de V ou F;
- Substituição;
- Encadeamento de condições;

And (E) / Condição V - Todas devem ser satisfeitas



p	q	p and q
V	V	V
V	F	F
F	V	F
F	F	F

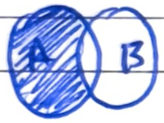
www.cadersil.com.br
CADERSIL

(União)

- OR (Ou) / Condição verdadeira → Apenas uma condição deve ser verdadeira.

p	q	p or q
V	V	V
V	F	V
F	V	V
F	F	F

- NOT : Operação de Negação
  Inversão do resultado lógico



p	~p
V	F

04. Estruturas de Repetição —— * Repete trecho ——

- Laços;
- Controle de fluxo;
- Malhas de repetição;
- Repetição;
- loop;

* Vantagem
  - Redução de linhas;
  - Compreensão facilitada;
  - Redução de erro;

* Condição de parada
  - Número de repetições pré-fixada;
  - Condição a ser satisfeita;

Estruturas:
- Enquanto _ faça
- Repita _ Até
- Para _ de _ até _ faça

05 Vetores e Matrizes

- Vetor: sequência de tipos de dados.
  - Uma variável dimensionada com tamanho pré-fixado.
  - n-valores;

* Vetor: [o o o o o o]

Matriz: uma tabela organizada em linhas e colunas no formato m x n, onde m - linhas e n - colunas.

L → o o o
I o o o
N o o o
H ↑
A COLUNA

- Coleção de variáveis;
- Contíguas em memória;
- Índices; (Permite navegar pela matriz)

06. O que são funções?

- Sub algoritmo; / Subprograma;
- Bloco;
- Método; (POO)
- Sub-rotina;

* Blocos de instruções (códigos) que realizam tarefas específicas.

- Decomposição do algoritmo;
- Modularização do problema;
- Identificados por nomes e parâmetros;

▷ Código mais claro e conciso;
Reutilização de instruções;

* Funções ▷ Altera o estado do programa.

- Definição;
- Nome;
- Invocação;
- Variável local; (Destruídas ao encerrar a função)

07. Instruções de entrada/saída

Dados → Processamento → Resultado

Def: Consiste na inserção e recebimento de dados do mundo real por meio de ação de alguma interface, seja teclado, mouse, arquivos, entre outros.

ENTRADA ▷

www.cadersil.com.br

CADERSIL

Saída: Impressão dos dados do mundo abstrato, digital por meio de ação de alguma interface.
- logo do sistema;

* Dois tipos de saída:
- Saída por interrupção; (Definida por periféricos)
- Saída programada;
  - Condicional; (Aguardo o dispositivo)
  - Incondicional;

Saída na programação ⟹ Código
Casos:
- Bem sucedida
- Erro de sintaxe ou outro
- Erro de programação
- Problemas com interface

11·03·25

4 LINGUAGENS DE PROGRAMAÇÃO

01. Introdução à linguagem de programação
02. Como um computador entende o programa
03. Característica de um programa
04. Análise de código
05. Paradigmas de programação

01. Introdução à linguagem de programação

- Introdução:
  - Dificuldades enfrentadas;
  - Fundamentos da computação;
  - O processo de pensamento;

▷ Pesquisas > Novos paradigmas

Evolução: Hardware (1ª) —▷ Software (2ª)

Capacidade Computacional — Limita

- 2º Guerra (39/45)
  - Investimento pesquisa;

* 3k A.C → 1º dispositivo de cálculo (Ábaco)

↓

- (1820) Instrumentos Computacionais

↓

- (1937) Conceito de software (Charles Babagge)

↓

- (1940) Alan Turing decifra Enigma; (9 matemáticos)

↓

- (1942 - 43) Transcrição de Ada Lovelace

↓

- (40 / 50) Projeto Dilab EVA Codificador de Voz
  Turing /

11/03/25

- 1943 - COLOUSSUS - programação por fios;
- 1952 - EDVAC - programa armazenado;

1975 — Intel 8080 ▷ Linguagem basic - Microsoft
1976 — Apple ▷
1977 — Apple 2 ▷

▷ **Apple sempre a frente!**

- 1949 - Assembly (1º linguagem)
  - Linguagem de Montagem;
  - Máquina;
  - Baixo nível;

= Problemas computacionais
- Cada era com um problema específico;

▷ Objeto de discussão que possui instruções passo a passo que são mais facilmente resolvíveis em ambiente computacional.

PROBLEMAS
- ① DECISÃO — LÓGICO (V/F), PERTECIMENTO.
- ② BUSCA — BINÁRIO, RELACIONAMENTO
- ③ OTIMIZAÇÃO — APERFEIÇOAR

① Linguagem de Programação: "Método padronizado composto por um conjunto de regras sintáticas e semânticas de implementação de um código fonte."

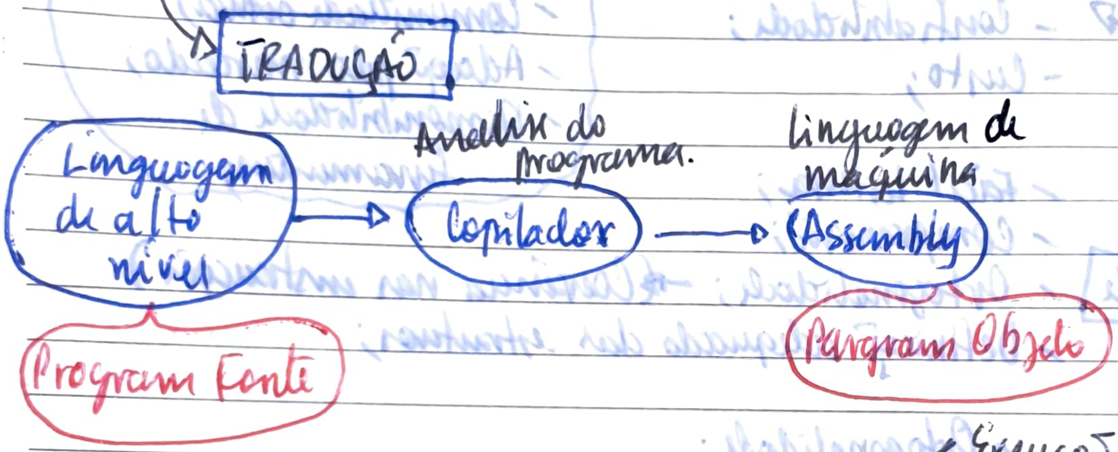
- Sintaxe;
- Operações;
- * SEMÂNTICA → ALGORITMO

CÓDIGO FONTE
- Tradução
- Interpretação

11 · 03 · 25

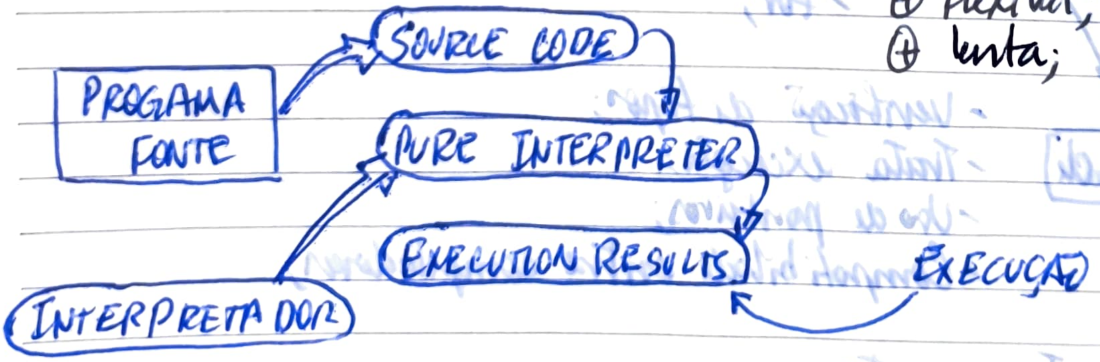
02. Como um computador entende o programa?

▷ PROGRAMA: "Amontoado de palavras ~~senão~~ se não for possível que o computador entenda!"



1. Tradução: - Geração do programa objeto;
   - Execução do programa objeto;
   - Execução ⊕ Rápida;
   - Programas menores;

2. Interpretação: programa fonte executado diretamente;
   ⊕ Flexível;
   ⊕ Lenta;



* Transpilação

11 · 03 · 25

03. Características de um programa

- Desenvolvimento de programas

DIRETRIZES →

- Legibilidade;
- Redigibilidade;
- Confiabilidade;
- Custo;

+ Atualização;
- IA;
- Comunidade ativa;
- Adoção pelo mercado;
- Disponibilidade de ferramentas;

Legibilidade

- Fácil leitura;
- Compreensão;
- Ortogonalidade; → (Coerência nas instruções)
- Definição adequada das estruturas;

Facilidade de escrita

Redigibilidade

- Ortogonalidade;
- Simplicidade de escrita;
- Suporte à abstração;
- Reuso do código;
- Expressividade;
  - Operador ++;
  - for;

Fazer o que for programado

Confiabilidade

- Verificação de tipos;
- Trata exceções;
- Uso de ponteiros;
- Compatibilidade entre compiladores;

Análise de impacto

Custo

Otimização dos gastos

- Treinamento;
- Codificação;
- Compilação;
- Execução;
- Infra-estrutura;

11 · 03 · 25

01. Análise de Código

- Analyzer
  - lexical;
  - syntax;
  - Semantic;

* Léxica → (token) → Elementos
  - Particionar;
  - Classificar;
  - Eliminar;
    - Caracteres espaços;
    - Comentários

  Elementos:
  - Identificadores
  - palavras reservadas
  - números
  - strings

* Sintática (forma)
  - Corretude do programa;
  - Atribuir uma estrutura;
  - Interligam os constituintes da sentença;

  - Padrão - Gramática:
    - Depende da linguagem de programação utilizada;

* Semântica: estudo do significado. → lógica do programa
  Incide sobre a relação entre significantes, como:
  - ~~??~~ palavras;
  - frases;
  - ~~???~~ sinais;
  - Símbolos;

+ difícil de achar!

11 · 03 · 25

05. Paradigmas de Programação

Def: Forma de resolução de problemas com diretrizes e limitações específicas de cada paradigma utilizando linguagem de programação.

CLASSIFICAÇÃO

- Orientação à objeto;
- Estruturado; (Estrutura de blocos aninhados)
- Procedural; (Chamadas sucessivas e procedimentos separados)
- Computação distribuída; (Funções executadas de forma independente)
- Funcional; (Instruções baseadas em funções)
- Lógico;

▷ Mais usadas: ① Orientação à Objeto
② Estruturado (Enfase em sequência)

② ESTRUTURADO:

* Conceitos:
  - Sequência;
  - Decisão; (Teste lógico)
  - Iteração; (Funções, laços, condições.)

Dado abstrato: pilha, grafo... lista.

- Utilização:
  - Problemas simples e diretos;
  - Aprender;
  - Problemas específicos e direto;

③ ORIENTAÇÃO À OBJETO

- Utilização de objetos e suas interações;
- Análogo ao mundo real;

Tudo no mundo real é um objeto.

* OBJETO: descrito por características específicas, Comportamentos e estados.

CADERSIL
www.cadersil.com.br

Classe: abstração de uma entidade de software.

O que tenho. Modelo → classificação

Sou capaz de fazer, escrever → comportamentos

como faço: Tampada, Destampada, Em uso } (Estado)

(Modelo) (Atributo)

(Comportamento) (Método)

(Estado)

Método - função associada a uma classe.

POO

Classe
- Alocação de Memória
- Operações associadas

Estruturada → Alocação em Memória + Operações associadas } Variável

Associados a uma classe e somente a uma.

- Reuso de código.

Pilares de O.O
- Herança
- Encapsulamento
- Polimorfismo
- Abstração

Classes

CLASSE MÃE → Comportamento e Estado gerais

Comportamento Geral

C. Filho → Comportamento e estado Específicos

C. Filha → Comportamento e estado Específicos

comportamento específico.

11.03.25

Primeiro contato com a programação

01. Algoritmos em portugol
02. Considerações finais
03. Material
04. Quiz

01. Algoritmos em portugol
- Portugol Studio on-line;
- Exemplo;

$\begin{cases} 1 = y \\ 10 = x \end{cases}$

- $\frac{y}{2}$ $\quad \frac{y}{2} \cdot (x+y)$

- $x+y$ $\quad \frac{1}{2} \cdot (10+1) = \frac{10}{2} \cdot \frac{1}{2} = \frac{11}{2} \quad \frac{10}{4} = \frac{5}{2}$

02. Considerações Finais
- Computional Thinking;
- lógica para desenvolvimento de programação de computadores;

CADERSIL
www.cadersil.com.br